VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

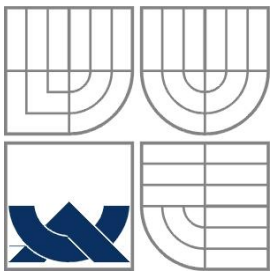
INTELIGENTNÍ WEBOVÝ PLÁNOVAČ PRÁCE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

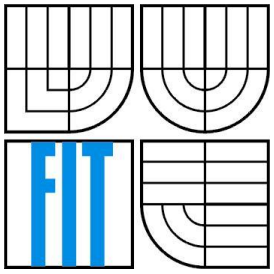
AUTOR PRÁCE
AUTHOR

MARTIN HALFAR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INTELIGENTNÍ WEBOVÝ PLÁNOVAČ PRÁCE

INTELLIGENT WEB PLANNER OF WORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN HALFAR
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ČERMÁK
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá využitím evolučních algoritmů při tvorbě informačního systému, jenž umožňuje inteligentně rozkládat zadanou práci mezi skupinu jeho uživatelů.
K řešení problému využívá hlavně genetické algoritmy, jenž jsou inspirovány evolučními procesy, probíhajícími v biologických systémech. Jednotlivá řešení problému označí za jedince v generaci a k procesu křížení připouští pouze nejpřizpůsobenější jedince.

Abstract

This bachelor's thesis deals with usage of Evolutionary algorithm while intelligently planning the distribution of work for a group of people. Especially, Genetic algorithm is addressed, which is one of techniques of evolutionary algorithm inspired with biological evolutionary processes.Every solution is branded as an individual of population and only the most adapted ones are allowed to reproduce.

Klíčová slova

Evoluční algoritmy, genetické algoritmy, plánovač práce, náhodný genetický drift.

Keywords

Evolutionary algorithm, genetic algorithm, planner of work, random genetic drift.

Citace

Halfar Martin: Inteligentní webový plánovač práce, bakalářská práce, Brno, FIT VUT v Brně, 2009

Inteligentní webový plánovač práce

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Čermáka Martina Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Halfar
20.5.2009

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Martinu Čermákovi za jeho odbornou pomoc, kterou mi poskytnul během vytváření práce

Obsah

Úvod

Úkolem této bakalářské práce je navrhnout systém, jenž bude akceptovat skupinu osob a na základě jejich aktuální vytíženosti mezi ně rozdělovat zadanou práci s využitím evolučních algoritmů.

Evoluční algoritmy patří do oboru inteligentních výpočtů (angl. softcomputing) a nabízí celkem čtyři techniky. Genetické algoritmy, jedna z nich, je pro řešení problému plánování práce nejvíce vyhovující, a proto bude pro řešení tohoto problému využita.

Genetické algoritmy vznikly v 70. létech minulého století a od ostatních technik se liší hlavně zavedením operátoru křížení jako primárního rekombinačního operátoru. Jejich největší výhodou je, že nehledají optimalizace slepě náhodným generováním nového řešení. Inspirací v Darwinově evoluční teorii nalezly způsob, jak usměrnit náhodné generování bodů k hodnotám blízkým optimálním.

Kapitola 1 této práce je lehkým úvodem do problematiky evolučních algoritmů. Popisuje jejich základní principy přirozený výběr, náhodný genetický drift a reprodukční proces a vysvětluje, proč se díky nim stávají v praxi tak mocným nástrojem k řešení složitých nebo jinak neřešitelných úloh, jakými jsou např. plánování práce či strojové učení.

Genetické algoritmy se k řešení úloh inspirovaly jednotlivými fázemi evolučních procesů probíhajícími v přírodě. V kapitole 2 nalezneme popis těchto fází a technik křížení, selekce a mutace, jimiž se je genetické algoritmy snaží napodobit.

Praktické využití těchto postupů k vytváření rozvrhu popisuje kapitola 3. Je zde uvedena charakteristika plánování práce a jednotlivé fáze plánování od zadání úkolů po jejich kódování do genetické informace jedinců.

Hlavním tématem kapitoly 4 je výčet požadavků, které jsou na plánovač práce v dnešní době kladeny. Určuje tím směr, jakým bude vývoj informačního systému dále veden.

Přímé navázání na kladené požadavky představuje kapitola 5, která představuje konkrétní stěžejní prvky plánovače práce, které budou sloužit v konečné implementaci jejich plnění.

Kapitola 6 se věnuje popisu způsobu řešení jednotlivých částí implementace plánovače. Nalezneme zde popis databáze, implementace algoritmu samotného plánovače a popis uživatelského rozhraní.

Závěr se pokusí zhodnotit přínos této práce a možné pokračování práce v budoucnosti.

1 Evoluční algoritmy

Evoluční algoritmy patří do skupiny stochastických optimalizačních algoritmů. Používají se k hledání globálního minima, obklopeného lokálními minimy.

Jejich největší síla a zároveň slabina je v jejich všeobecnosti. Proto se jejich využití omezuje ve většině případů na problémy, k nimž je velice složité anebo nemožné sestrojit specializované algoritmy (v praxi se používají pro navrhování obvodů, plánování výroby, strojové učení atd.). Jejich další výhodou je, že pracují pouze s funkcemi samotnými, nikoli s derivacemi optimalizační funkce.

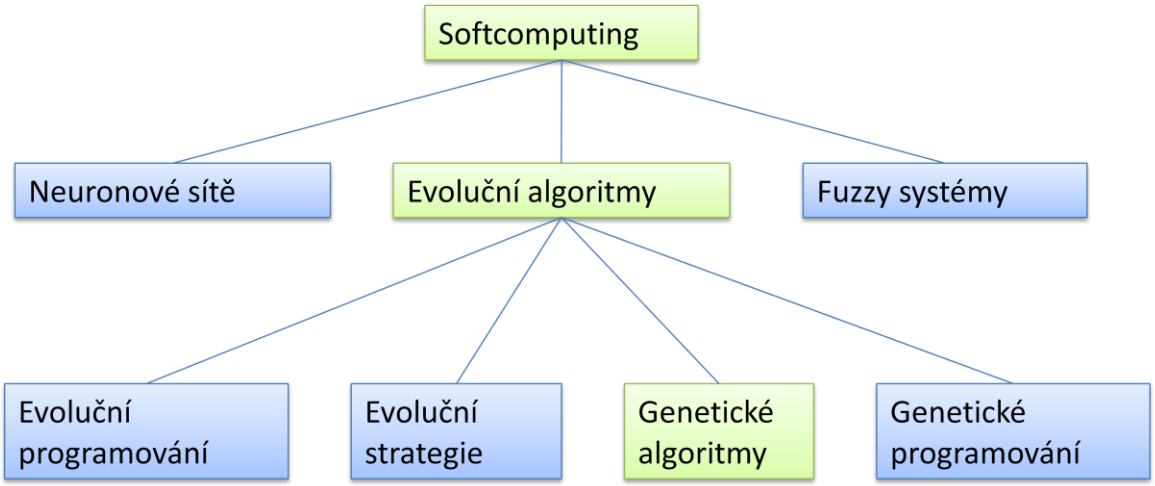
K hledání nejlepšího možného řešení využívají evoluční algoritmy znalosti nabyté z přírodních věd a to především z Darwinovy evoluční teorie. Ze základních rysů klasického Darwinismu přebírají tyto tři komponenty:

1. **Přirozený výběr**, proces, do kterého vstupují jedinci z generace s nejvyšší funkcí výhodnosti (tzv. fitness).
2. **Náhodný genetický drift**, ve kterém náhodné události v životě jedinců ovlivňují celou populaci. V přírodě lze nalézt podobnost v náhodné mutaci či nečekaném úmrtí jedince s nevyšší fitness, tj. s největší pravděpodobností vstoupit do reprodukčního procesu.
3. **Reprodukční proces**, v rámci kterého se z rodičů vytvářejí potomci. Genetická informace potomků se při tom skládá pouze z genetických informací obou rodičů.

Základním pojmem evolučních algoritmů je populace chromozómů. V těchto chromozómech je zakódováno řešení optimalizačního problému. Chromozóm zakódovaný takovým způsobem, že jej lze jednoznačně označit za nejlepší možné řešení daného optimalizačního problému, bude mít nejvyšší fitness. V dané populaci tedy do procesu reprodukce vstupují rodičové s nejvyšší fitness a následný potomek projde s určitou pravděpodobností mutací, která v přírodě pomáhá najít takové řešení, jež se v populaci zatím nevyskytuje, ale zato může vést k vyššímu fitness.

Evoluční algoritmy patří v současnosti mezi základní nástroje moderní informatiky v případech hledání řešení v extrémně složitých situacích, kdy použití standardních deterministických metod založených na technikách úplného prohledávání není možné aplikovat. Za velmi efektivní přístup jsou označovány v případech, kdy nepotřebujeme nejoptimálnější řešení problému, ale vystačíme si i s kvalitním suboptimálním řešením.

Evoluční algoritmy patří do oboru inteligentních výpočtů, softcomputing (viz obrázek 1) a nabízí celkem čtyři techniky: evoluční programování, evoluční strategie, genetické programování a genetické algoritmy. Poslední zmíněná technika poslouží k hledání optimálního řešení problému plánování práce.



Obrázek 1. Základní rozložení oboru softcomputing

2 Genetické algoritmy

2.1 Úvod

Genetické algoritmy v současné době patří k nejčastěji používaným evolučním optimalizačním algoritmům. Jejich hlavním rozlišovacím znakem je použití operátoru křížení (crossover operator) jako primárního rekombinačního operátoru. Hlavní odpůrci genetických algoritmů považují tento operátor za „rozbíjení“ bloků bitů a aplikují jej stejně jako mutaci s velmi malou pravděpodobností.

Oproti algoritmům stochastické optimalizace genetické algoritmy nehledají optimalizace slepě náhodným generováním nového řešení. Genetické algoritmy nalezly způsob jak usměrnit náhodné generování bodů k hodnotám blízkým optimálním.

Darwinova evoluční teorie, jíž se genetické algoritmy inspirují, se zakládá na teorii přirozeného výběru, podle které přežívají pouze nejlépe přizpůsobení jedinci populace. Reprodukcí dvou jedinců s vysokým fitness dostáváme potomky, kteří budou s vysokou pravděpodobností dobře přizpůsobení na úspěšné přežití. Při podrobné analýze však dospějeme k závěru, že samotné působení reprodukce není dostatečně efektivní. Pro vylepšení tohoto procesu je zapotřebí přidat tzv. mutace. Ty náhodným způsobem ovlivňují genetickou informaci potomků buď negativním, anebo pozitivním způsobem. Pomáhají tudíž hledat i takové nové cesty k nalezení optima, kterými nedisponovala genetická informace jejich rodičů.

2.2 Základní pojmy genetického algoritmu

Chromozóm (jedinec/individuum) kóduje řešení a je reprezentován většinou binárním vektorem (řetězcem) konstantní délky k. Populace P je potom množina takovýchto chromozómů

(1) $$P = \{ \alpha_1, \alpha_2, ..., \alpha_n\} \subseteq \{a, b, ... \}^k$$

Populace *P* obsahuje *n* chromozómů. Každý chromozóm $\alpha \in P$ je ohodnocen fitness, která se interpretuje jako zobrazení chromozómů na kladná reálná čísla.

(2) $$F:P \rightarrow R_+$$

Naší úlohou bude pak hledat globální minimum této funkce nad množinou $\{a, b, ...\}^k$.

(3) $$\alpha_{opt} = arg\ min\ f(\alpha)$$

V biologické terminologii chromozóm α reprezentuje genotyp organizmu (neboli způsob kódování organizmu), zatímco funkční hodnota $f(\alpha)$ reprezentuje fenotyp organizmu (neboli míru úspěšnosti organizmu). Protože hledáme minimum účelové funkce, chromozóm je úspěšnější tím víc, čím nižší je jeho funkční hodnota.

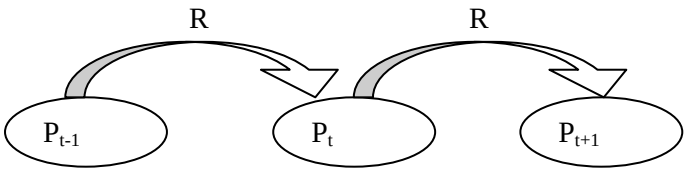
Proces reprodukce chromozómů začíná selekcí (kvazináhodným výběrem dvou chromozómů v závislosti na jejich fitness - jedinci s vyšším fitness mají větší pravděpodobnost být vybráni) a výsledkem je, že se původní dva chromozómy α_1 a α_2 zreprodukují na dva chromozómy α_1' a α_2'.

(4) $$(\alpha_1' \ a \ \alpha_2') = O_{reprodukce}(\alpha_1 \ a \ \alpha_2)$$

Operátor reprodukce obsahuje dvě části: křížení a mutaci. Tento operátor má stochastický charakter, obě jeho části se vykonávají jen s určitou pravděpodobností.

Evoluci můžeme charakterizovat jako postupnou rekurentní změnu populace (viz obrázek 2). Nechť P_t je populace chromozómů v čase *t*, potom v následujícím čase $t+1$ je populace P_{t+1} určená pomocí „reprodukčního“ R-operátoru.

(5) $$P_{t+1} = R(P_t)$$



Obrázek 2. Evoluce populace P v čase t

2.3 Selekce

Operátor selekce vytváří novou populaci $P(t+1)$ výběrem jedinců s možným opakováním ze staré populace $P(t)$. Výběr musí dostatečně upřednostňovat jedince s vyšší fitness, ale na druhou stranu musí udržovat dostatečnou různorodost, aby nehrozilo nebezpečí předčasné konvergence. Selekce může probíhat několika způsoby. Mezi nejběžněji používané patří výběr pomocí rulety (proporcionální selekce), turnajová selekce nebo exponenciální uspořádání.

2.3.1 Výběr pomocí rulety

Jde o nejčastěji využívaný typ selekce. Pravděpodobnost výběru i-tého jedince je dána následující pravděpodobností:

(6) $$p_i = \frac{f_i}{\sum_{j=0}^{N} f_j}$$

Cílem selekce z množiny rodičů-chromozómů je dát větší šanci jedincům s vyšší fitness. Závislost pravděpodobnosti výběru na velikosti fitness však může působit v jistých situacích i negativně. Vyskytne-li se v populaci jedinec s relativně vysokým fitness f_i, je populace postupně nahrazena tímto jediným chromozómem. Tomuto trendu se dá předejít úpravou původní fitness funkce (tzv. škálováním). Nejčastěji používané dvě techniky pro to jsou:

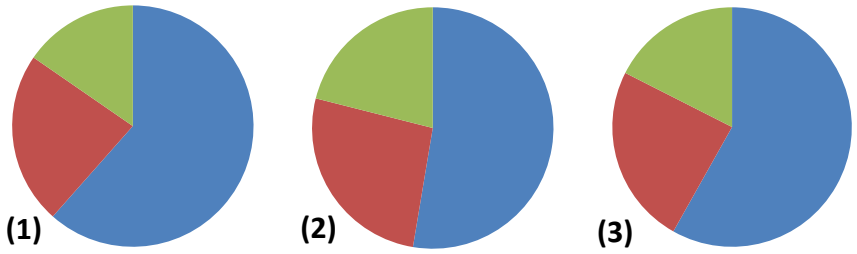
1. Komprimace fitness funkce (windowing):

   $f'(i) = f(i) + \beta^t$, kde je nejhorší fitness v aktuální t-generaci

2. Sigma škálování:

   $f'(i) = max(f(i) - (\langle f \rangle - c * \sigma_f), 0.0)$, kde c je konstanta, obvykle 2.0, a $\langle f \rangle$ je střední hodnota fitness funkce a σ_f je rozptyl hodnot před selekcí

Tuto technika lze přirovnat k házení kuličky do rulety (od toho ten název). Na grafech na obrázku 3 můžeme vidět, že ve výsledku po aplikování obou technik dojde pouze ke zredukování rozdílu mezi výsečemi. Tím bude zajištěna dostatečná diverzita v populaci.



Obrázek 3. (1) Výběr pomocí rulety, (2) Windowing, (3) Sigma škálování

2.3.2 Exponenciální uspořádání

Pravděpodobnost výběru v populaci je rozložena s exponenciální závislostí. Index N představuje index nejlepšího jedince v populaci. Pravděpodobnost výběru i-tého jedince je dána:

(7) $$P_{exp}(i) = \frac{1 - e^{-i}}{c}$$

(8) $$i \in \{1, 2, ..., N\}$$

Parametr c se volí v rozsahu $0 < c < 1$. Tento selekční algoritmus patří z uvedené skupiny k nejlepším.

2.3.3 Turnajová selekce

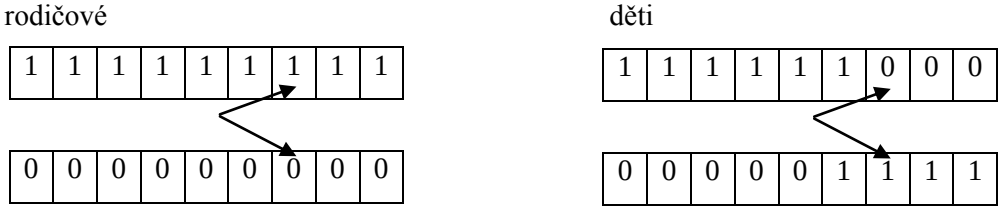
Výsledky tohoto algoritmu jsou velmi podobné předchozí metodě. Mezi jeho výhody patří především absence požadavku na setřídění populace a jednoduchost vlastní selekce.
Princip je jednoduchý, podobný tomu na kolbišti. Z *N* jedinců populace je vybráno *t* jedinců. Do další generace postupuje nejlepší z těchto *t* jedinců. Celý postup opakujeme do chvíle, kdy máme vybráno N jedinců do nové populace.

2.4 Křížení

Použití operátoru křížení (crossover operator) jako primárního rekombinačního operátoru je hlavní charakteristikou genetických algoritmů. Bývá prováděn s pravděpodobností p_c, většinou 70%, čímž je zajištěno, aby se část jedinců generace předchozí replikovalo bez nutnosti výměny genů.

2.4.1 Jednobodové křížení

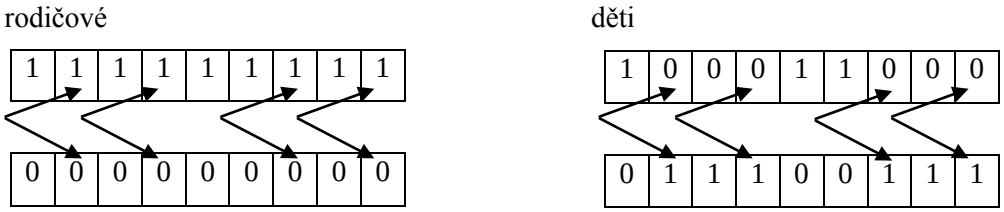
První z možností aplikace operátoru křížení je jednobodové křížení. Dochází při něm od určitého místa v genetické informaci rodičů k výměně jejich genů.



Obrázek 4. Jednobodové křížení

2.4.2 Vícebodové křížení

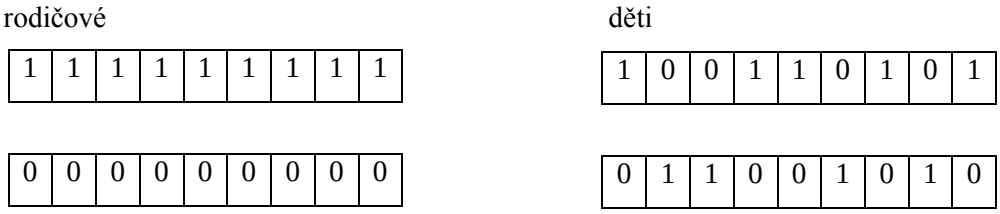
Je analogií jednobodového křížení pouze s jednou modifikací. Namísto jednoho bodu je jich u této metody zvoleno více, ohraničujících celé bloky genetické informace pro výměnu.



Obrázek 5. Vícebodové křížení

2.4.3 Uniformní křížení

S pravděpodobností p_u dochází k výměně jednotlivých genů obou rodičů. U této metody se již tedy neurčují žádné body a navíc u ní nedochází tak často k předčasné konvergenci algoritmu. Používá se proto při řešení složitých vícerozměrných funkcí s mnoha lokálními extrémy.



Obrázek 6. Uniformní křížení

2.5 Mutace

Je součástí procesu reprodukce a jsou pro něj zdrojem nových informací. Dochází k ní s pravděpodobností p_m. Příliš častá mutace způsobuje nestabilitu vývoje a naopak při příliš nízké pravděpodobnosti mutace nepřináší dostatek informací pro další vývoj.

3 Plánování práce

3.1 Úvod

Všechny firmy potřebují efektivně spravovat využití svých zaměstnanců a průběžně kontrolovat jejich výkonnost. Lidský faktor při navrhování takového rozvrhu práce, umožňuje jen velmi těžko u větších systémů objektivní pohled na všechny zaměstnance ve firmě. Navíc v případě jakýchkoli změn by nutná aktualizace rozvrhu trvala dlouho a docházelo by tudíž i ke zbytečnému plýtvání prostředky a možným chybám.

Metodou prohledávání všech možných řešení by nám navržení rozvrhu práce trvalo také neúměrně dlouho. Zato evoluční algoritmy se obecně používají k řešení problémů, ke kterým chybí nebo by bylo složité sestrojit specializovaný program. Stávají se tak mocným nástrojem a poskytují nám relativně rychlou cestu k dosažení alespoň suboptimálního řešení. V tomto případě sestrojení takového rozvrhu práce, který bude efektivní, nebude zbytečně plýtvat prostředky firmy a bude v případě změn aktualizovatelný.

3.2 Charakteristika plánování práce

Práce zahrnuje množinu úkolů, které musí být zpracovány. Pro každý takový úkol je ve firmě přiřazen jeden či více specialistů. Aby mohla být práce úspěšně dokončena, musí být všechny její úkoly zpracovávány v daném pořadí.

Práce	
Úkol 1	Úkol 2

Obrázek 7. Rozklad práce na úkoly 1 a 2. Úkol 1 musí být dokončen před započetím úkolu 2. Práce bude dokončena, jakmile budou dokončeny oba dva úkoly.

Každý zaměstnanec může v daný okamžik pracovat maximálně na jednom úkolu. Různé úkoly stejné práce se nemohou zpracovávat současně.

Ve všech firmách existují lidé, kteří se snaží profitovat na úspěších či zručnosti druhých. Cílem je tedy zvýhodnit a tím motivovat zaměstnance, kteří se snaží a je to na jejich výsledcích vidět. Pro člověka nejlepší motivací pro vyšší výkonnost jsou peníze. Systém proto bude v závislosti na pracovní efektivitě navrhovat prémie jednotlivým zaměstnancům a bude z ní také vycházet při rozhodování o přidělování práce.

Cílem kvalitního plánování práce je stihnout za jeden pracovní den co nejvíce prací s ohledem na jejich prioritu a zatěžovat při nich v první řadě pracovitějšího zaměstnance.

Pracovitost zaměstnance se měří jako průměrná doba nutná na zvládnutí daného typu úkolu.

3.3 Fáze plánování práce

3.3.1 Zadání úkolu

Zadání je tvořeno dvěma tabulkami. První tabulka každému zaměstnanci ve firmě specifikuje jeho efektivitu pro každý úkol. Obsahuje tedy seznam všech zaměstnanců a jejich specializaci. Podle specializace každého z nich poté systém při pánování práce doplňuje seznam úkolů, které již daný zaměstnanec dokončil a dobu k tomu potřebnou. Druhá tabulka obsahuje seznam prací, které firma poskytuje a jejich rozklad na typy úkolů v přesně daném pořadí. Jelikož bude systém rozhodovat o rozdělování úkolů jednotlivým zaměstnancům, musí být názvy prací a úkolů standardizovány. V případě, že firma nabízí omezený sortiment služeb, bylo by optimálnější tuto tabulku příliš často neměnit. Zamezilo by se tím vnesení chyby lidským faktorem a navíc by se stal systém pro zaměstnance maximálně transparentní a motivující. Příklad takovýchto dvou tabulek vidíme na obrázku 8.

	Typ úkolu 1	Typ úkolu 2
Maťa	ND	3
Novák	1	ND
Sobotka	ND	2,5

A

	Typ úkolu	Typ úkolu
Práce 1	Typ úkolu 2	Typ úkolu 1
Práce 2	Typ úkolu 1	Typ úkolu 2
Práce 3	Typ úkolu 2	Typ úkolu 1

B

Obrázek 8. Tabulka A znázorňuje ohodnocení zaměstnanců (doba potřebná na dokončení jednotlivých typů úkolů) Na tabulce B vidíme rozklad práce na jednotlivé úkoly (pořadí úkolů). Zkratka ND u zaměstnance znamená, že nemá specializaci pro tento typ úkolu.

3.3.2 Tvorba rozvrhu práce

Pro tvoření genotypu je možné použít různé typy kódování. První varianta, kterou zde uvedu, kóduje po úkolech.

3.3.2.1 Kódování po úkolech

Algoritmus náhodně vybere práci z tabulky B na obrázku 8. V pořadí první typ úkolu této práce vloží do výsledného řetězce úkolů a poznačí si, že tento úkol už zpracoval. Pokud příště algoritmus znovu náhodně vybere stejnou práci, zpracuje následující úkol. Tyto činnosti se opakují do chvíle, než bude rozvrh plný, nebo bude mít u tabulky B na obrázku 8 poznačeno, že již zpracoval všechny její úkoly. Příklad výsledného řetězce úkolů můžeme vidět na obrázku 9.

Typ úkolu 2 (Práce 1)	Typ úkolu 2 (Práce 3)	Typ úkolu 1 (Práce 3)	Typ úkolu 1 (Práce 1)	Typ úkolu 1 (Práce 2)	Typ úkolu 2 (Práce 2)

Obrázek 9. Příklad kódování po úkolech

V dalším kroku algoritmu systém již sestaví výsledný rozvrh práce. Jednotlivé typy úkolů bude postupně přiřazovat zaměstnancům dle jejich výkonnosti. Výsledný rozvrh práce by vypadal jako na obrázku 10.

	6:00	6:30	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00
Maťa	Typ úkolu 2										
Novák	Typ úkolu 1						Typ úkolu 1		Typ úkolu 1		
Sobotka	Typ úkolu 2					Typ úkolu 2					

Obrázek 10. Příklad rozvrhu práce při kódování po úkolech

3.3.2.2 Kódování po pracích - permutační kódování

Jak již název napoví, tentokrát algoritmus práci dál nedělí na jednotlivé úkoly. Ale do výsledného genotypu zakóduje práce jako celek, stejně jako můžeme vidět na obrázku 11 (příklad opět vychází z tabulek na obrázku 8).

Práce 1	Práce 3	Práce 2

Obrázek 11. Příklad kódování po pracích

Při tvorbě výsledného rozvrhu už tedy nedochází k tak velkým prodlevám mezi jednotlivými úkoly stejné práce (viz obrázek 11) a při sestavování složitějších rozvrhů by proto měl být tento způsob výhodnější. Již na tomto malém případu vidíme, že u Nováka jsme „ušetřili“ půl hodiny práce. Obecně platí, že permutační kódování dává lepší výsledky.

	6:00	6:30	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00
Maťa	Typ úkolu 2										
Novák	Typ úkolu 1					Typ úkolu 1		Typ úkolu 1			
Sobotka	Typ úkolu 2					Typ úkolu 2					

Obrázek 12. Příklad kódování po pracích

4 Požadavky na plánovač práce

V současné době se ve firmách preferuje především rozdělování práce mezi zaměstnance na základě úsudku nejkompetentnějšího člověka, který má přehled o obtížnosti jednotlivých úkolů. Není žádným překvapením, že se jím často stávají ti, kdo si prací, o níž mají rozhodovat, prošli. Naprogramovat jediný systém, který by takto zkušeného člověka zastoupil úplně ve všech odvětvích lidské činnosti lze považovat za extrémně náročný úkol.

Východiskem takové situace lze označit relativně malý informační systém, postavený na míru každému oboru činnosti člověka. Takovýto informační systém by se učil hned ze začátku od zkušenějších a postupem času by se jim svými schopnostmi začal blížit. Prostředky současné doby zatím nestačí na vytvoření absolutně chytrého, neomylného systému. Proto je nutné i přes veškerou snahu počítat s možností chyby a poskytnout správcům nezbytný přehled a přístup k nástrojům k její nápravě.

4.1 Jednoduchost

V dnešní době je hodně kladen důraz na jednoduchost ovládání, aby se tak ušetřily náklady za případné školení či počáteční ztráty z důvodů neznalosti personálu. Mnou naprogramovaný systém se bude snažit předvídat nejčastěji prováděné operace, ty do maximální míry zjednodušit a nezatěžovat uživatele zbytečnými informacemi. Ovládat ho tedy nebude problém pro žádného zaměstnance. Těžší prvky jeho správy budou běžným zaměstnancům skryty a zůstanou na správcích.

4.2 Kompaktnost

Pokrytí bezdrátového připojení k internetu již není žádným problémem a ceny hardware klesly tak nízko, že si téměř každá firma může dovolit vybavit své zaměstnance levnými příručními zařízeními. Dá se proto předpokládat, že tento trend bude pokračovat a software náročný na procesorový čas se bude přesouvat na centrální servery. Uživatelé pak budou mít k dispozici klienta, který bude zprostředkovávat přístup. V tom to duchu bude pojat vývoj tohoto plánovače.

Všechny náročné výpočty, nezbytné ze samotné podstaty evolučních algoritmů, zůstanou na serveru, k němuž se uživatelé budou připojovat přes nevýkonná kapesní zařízení typu mobil či PDA. Tento plánovač práce se tedy bude hodit spíše firmám, jejichž zaměstnanci pracují během pracovní doby na více místech. Odpadne jim tak spousta práce se synchronizací jejich práce, přibude centrální správa, možnosti monitorování aktivity zaměstnanců, objektivní prostředky k jejich ohodnocení. I přesto však zůstane zaměstnancům jejich volnost pohybu či možnost volby pracovního nasazení.

4.3 Správa

Vzhledem k určení tohoto informačního systému menším firmám, nesmí být jeho správa náročná. Aby systém moc nemátl své uživatele, budou mu některá pokročilá nastavení v uživatelském rozhraní spravování systému skryta. Přesto zůstane možnost systém ladit pomocí konfiguračního souboru.

Uživatelské rozhraní tak nabídne jeho uživateli pouze přehled nad svými běžně potřebnými částmi.

5 Prvky plánovače práce

5.1 Hodnocení úkolů

Nejideálnější stav při hodnocení úkolů by byl nadhled nad jejich obtížností. Potom by se lehčí úkol, přidělený zkušenému zaměstnanci dal ohodnotit větší sumou peněz na hodinu, protože ji zkušenější pracovník vždy zvládne rychleji. Jak ale rozhodnout o obtížnosti práce, když o ní nic nevíme? Faktory, které se na práci hodnotí, by se daly v běžném životě zredukovat na hlavní dva: počet dní do nejzazšího termínu dokončení a její priorita, která potřebu dokončit ji ještě více umocní.

V podstatě nikdy nebudeme hodnotit práci jako takovou, ale vždy hodnotíme pouze zaměstnance za to, jak rychle ji zvládnul dokončit v porovnání s ostatními.

Pokud tedy chceme úkoly spravedlivě rozdělit, musíme mít přehled hlavně nad výkonností firemních zaměstnanců.

5.2 Specializace úkolů

Každý zaměstnanec má svou kvalifikaci, s níž do firmy přišel a ta ho opravňuje plnit určitý repertoár úkolů. Jak tyto úkoly od sebe odlišit, aby je plánovač rozeznal?

Každý úkol lze nějak slovně popsat. Můžeme toho využít a pro každou množinu podobných úkolů vytvoříme jednu specializaci. Každý zaměstnanec, který potom na daném úkolu může pracovat, bude mít přiřazenu stejnou specializaci.

Jelikož může jeden zaměstnanec pracovat na větším množství prací, má tudíž i více specializací, o něž se dělí s větší skupinou spolupracovníku. Dostáváme se tak k nejednoznačnosti při rozdělování práce, kterou nám pomohou vyřešit právě evoluční algoritmy.

Při nejefektivnějším využití plánovače by se při prvním startu systému zadaly všechny specializace a přiřadily by se k jednotlivým zaměstnancům. Následně při přidávání nového úkolu do systému by ho pak stačilo ke specializaci z dané množiny a tím by byla určena skupina zaměstnanců, kteří na něm mohou pracovat.

5.3 Typy úkolů

Specializace úkolů nám však pomáhá vyřešit pouze část problému s rozdělováním práce. Co se bude dít, pokud v rámci jedné specializace lze dělat více podobných úkolů? Specializaci lze chápat spíše jako určení profese člověka, kdežto typ úkolu je určitější a vztahuje se více k samotnému úkolu. Při vypočítávání hodnocení zaměstnanců a při vedení statistik je proto lepší jej využít.

Oproti specializaci navíc nemusí každý úkol mít svůj typ. Úkoly jsou však bez typu rozdělovány mezi zaměstnance rovnoměrně a nejsou tak využity všechny výhody tohoto plánovače. Zaměstnanec je hodnocen na základě počtu provedení a průměrné délky jednotlivých typů úkolů v porovnání se spolupracovníky, kteří již na tomto typu úkolu také pracovali.

Případy, kdy není třeba brát v úvahu hodnocení zaměstnance, lze vyřešit právě nezadáním typu úkolu. Může se jednat buď jednorázové úkoly, anebo takové, které se objevují sporadicky a bylo by zbytečné je zavádět do systému a jejich statistiky ukládat v databázi.

5.4 Hodnocení zaměstnanců

Zaměstnanci jsou hodnoceni kvůli určení výše odměny, která se následně mění v motivaci. Jak ale hodnotit člověka tak, aby byl opravdu motivovaný, ale ne deprimovaný?

Když je zaměstnanci přidáno na platu, je potom šťastnější a v práci produktivnější. Pokud se mu naopak z platu ubere, vezme si to většinou osobně a nepodává výsledky, jakých by byl schopen.

Při hodnocení zaměstnanců bych tedy zvolil variantu, kdy člověku k základnímu platu za každý provedený úkol připočítám systémem navrženou sumu, anebo sumu správcem upravenou.

Správce bude mít možnost korigovat finanční ohodnocení zaměstnance mimo jiné také kvůli eliminaci průtahům, které si zaměstnanec ani nemusel způsobit sám, jako je např. závada na firemním zařízení, či dopravní kalamita při firemní cestě.

Zvoleným systémem hodnocení zaměstnanců se snažím o efektivní rozdělování úkolů v první řadě mezi nejrychlejší zaměstnance. Pokud by však byl systém špatně vyvážený, hrozilo by, že někteří zaměstnanci ve snaze vyhnout se práci budou pracovat čím dál méně a systém jim tím pádem bude přidělovat také méně práce. Tento stav by byl nežádoucí. Byl by velice nespravedlivý pro tvrdě pracující, obzvláště pokud by takoví zaměstnanci pobírali stejný. Proto neprobíhá výpočet hodnocení jen na základě délky odpracovaných úkolů jedinců, ale na základě porovnávání s jinými.

Jelikož budeme mít k dispozici nástroje, jako jsou specializace úkolů a typy úkolů, nebude problém spočítat si průměrnou délku vykonávání daného úkolu i u jiných zaměstnanců a porovnávat je mezi sebou. Pokud se budou výsledky navzájem hodně lišit, systém u jedince rozhodne o snížení prémií, anebo o snížení sazby za odvedenou práci.

6 Implementace

6.1 Databáze

Každá firma plní zakázky, které se skládají z více úkolů. Každý takový úkol vyžaduje určitý čas člověka s jistou kvalifikací. Existují takové, u kterých se dá dopředu předpokládat jejich délka, a v rámci firmy se často opakují. Je tudíž možné dle nich monitorovat efektivitu určité skupiny zaměstnanců, kteří jsou pro danou práci kvalifikovaní.

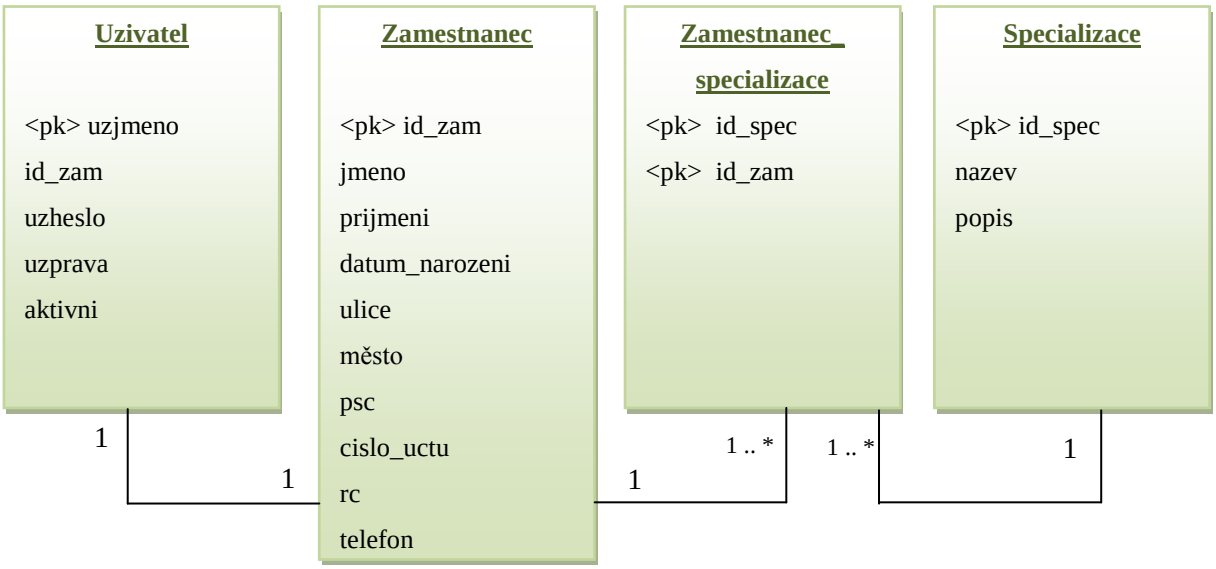
Úkoly, z nichž se skládají jednotlivé práce, musejí být provedeny v přesně daném pořadí zaměstnancem s potřebnou kvalifikací. Práce je prohlášena za hotovou až v případě, že budou dokončeny všechny jejich úkoly.

Zaměstnanec může mít libovolný počet specializací (viz. obrázek 13) a pracovat tak na větší množině typů úkolů.

Jeden úkol musí mít jen jednu specializaci a může mít nanejvýš jeden typ úkolu.

Návrh databáze počítá s jedenácti tabulkami:

- uzivatel
- zamestnanec
- zamestnanec_specializace
- specializace
- zamestnanec_typ
- typ_ukolu
- zamestnanec_ukol
- ukol
- se_sklada_z
- prace
- zamestnanci



Obrázek 13. Část ER diagramu

6.2 Informační systém

S informačním systémem, jehož ovládání je přímočaré a intuitivní se jeho uživatel rychleji sžije. Obzvlášť to platí pro plánovač práce, u kterého se předpokládá dlouhodobé využívání.

I přes snahu o co největší zjednodušení správy je web stále rozčleněn do relativně velkého množství stránek. Pro lepší orientaci v této struktuře je na každé stránce přítomna kontextová navigace, nabízející uživateli odkazy na stránky, které jsou vzhledem k aktuální pozici na webu jeho nejpravděpodobnější destinací.

K implementaci informačního systému tohoto inteligentního webového plánovače práce jsem použil kombinaci skriptovacího jazyku PHP ve spolupráci s databázovým jazykem MySQL. Oba tyto jazyky jsou veřejností velice oblíbené nejen pro jejich cenu, ale hlavně proto, že nabízí podobný komfort jako jejich placená konkurence.

6.2.1 Přihlašování do systému

Vzhledem k povaze zamýšleného systému jsou uživatelům všechny jeho stránky přístupné pouze pod heslem. Celkem existují dva typy uživatelů s různými oprávněními: zaměstnanec a poweruser.

6.2.1.1 Poweruser

Po přihlášení ho přivítá hlavní stránka s obecnými informacemi o plánovači. Hlavní navigace v horní části stánky se skládá z pěti odkazů. V sekci Rozvrhy si můžeme prohlížet rozvrhy jednotlivých zaměstnanců. Id zobrazovaného zaměstnance poweruser volí kliknutím v select boxu *„Zvolte zaměstnance“* a datum, ze kterého dne se má rozvrh zobrazit, zadává v poli *„Zadejte datum rozvrhu“*. Pokud je zadána taková kombinace data a id zaměstnance, že pro daný den neexistuje žádný rozvrh, vypíše systém upozornění *„Zaměstnanec nemá v tento den žádný rozvrh“*. V opačném případě poweruser uvidí výpis úkolů pro daný den v pořadí, v jakém byly plněny.

O každém splněném úkolu se v tomto výpisu zobrazí informace o jeho specializaci, prémiích, které za ně zaměstnanec dostal, původní odhad délky práce, konečná délka trvání úkolu, začátek plnění, datum a čas splnění úkolu a také stav, který může nabývat třech hodnot: *„dokončen“*, *„probíhá jeho plnění“* a *„čeká na zpracování“*.

Toto zobrazování nepoužívá evolučních algoritmů ke generování nových rozvrhů. Zobrazuje jen již dokončené úkoly anebo úkoly rozpracované – tedy takové, které jsou již přiřazeny ke zvolenému zaměstnanci v tabulce *zamestnanec_ukol*.

Rozpracované úkoly jsou ty, ke kterým je aktuálně zaměstnanec přihlášen a jejich stav je označen jako *„probíhá jejich plnění“*. Jelikož v tomto případě zatím není znám čas dokončení, jsou položky konec a délka úkolu výpisu takovýchto položek v rozvrhu označeny stejně.

Generování nového rozvrhu na určité datum v tomto případě nedělá žádné změny v databázi. Pouští jen algoritmus s aktuálními informacemi v databázi a poweruser se tak může informativně podívat na vytíženost zaměstnanců, případně jestli má pro daný den dostatek práce pro všechny specializace. Mimo to tyto výpisy mohou posloužit k ověření správné funkčnosti generování rozvrhu.

K dispozici má poweruser dále i sekci zaměstnanci.

Na výpisu ihned uvidí jejich aktuální pracovní vytížení – sloupec „stav“, který nabývá hodnot *„plní úkol“* nebo *„bez úkolu*. Orientačně se tak dá určit pracovní nasazení jednotlivých zaměstnanců.

Pokud bude chtít poweruser změnit specializaci nebo typy úkolů zaměstnance, klikne na odkaz upravit. Obě tyto vlastnosti se nastavují v select boxu s možností hromadného výběru. Vidí proto okamžitě, jaké specializace či typy úkolů jsou vybrány. Pokud se nám změny budou líbit, klikneme na

tlačítko *„Uložit změny“*, pokud se budeme chtít vrátit zpět k původním změnám, klikneme na odkaz *„Znovu načíst informace z databáze“*, který směřuje na tutéž stránku a v podstatě provede opět ty samé dotazy na databázi jako při první návštěvě této stránky. Jde o bezpečnější cestu obnovy stránky, než by bylo samotné stisknutí tlačítka F5 na klávesnici, které by mohlo vyústit k nechtěnému opětovnému odeslání formuláře. Do předešlého menu se dostaneme kliknutím na odkaz *„Zpět na seznam zaměstnanců“* pod tímto odkazem.

Když přejdeme do další sekce plánovače, uvidíme seznam všech prací v systému. Můžeme se zde přesvědčit o stavu jednotlivých prací, který je buď *„nedokončena“*, anebo *„dokončena“*. K podrobnějším výpisům se dostaneme přes odkaz *„detail“*, který nám umožní shlédnout i jednotlivé úkoly dané práce. Opět zde vidíme i stav plnění úkolu, který má tři stupně (*„dokončený“*, *„čeká na zpracování“*, *„probíhá jeho plnění“*).

Přes odkaz *„upravit“* při výpisu prací se dostaneme k podobnému výpisu s tím rozdílem, že tentokrát nám bude umožněno tyto údaje upravovat, mazat a přidávat.

Po přidání nové práce je nezbytné přidat k ní ještě alespoň jeden úkol. U práce se totiž nevolí její specializace. Ta zůstává doménou jednotlivých úkolů. Pro přidání úkolu k již přidání práce postupujeme vždy stejně: při upravování práce pod tlačítkem *„Uložit změny“* najdeme odkaz na *„přidat úkol“*.

Po správném vyplnění údajů, nutným k přidání záznamu, a odesláním kliknutím na *„přidat“*, nás vždy systém informuje o výsledku změn v databázi v horní části obrazovky. V případě, že změny proběhly úspěšně, se údaje ve formulářích, které přidávají záznamy, navíc vymažou, aby nedocházelo k matení uživatelů, zdali je již záznam uložen anebo ne. S nesprávnými hodnotami ve formuláři jej nebude dovoleno odelst.

V případě, že jsme zapomněli přidat do systému novou specializaci anebo typ úkolu, jsou na stránce přítomny odkazy, vedoucí přímo na přidávání nových specializace či typů úkolů.

Poslední položkou hlavní navigace je sekce předvolby. Poweruser je uživatel s nejvyššími oprávněními a má právo tato nejvyšší oprávnění přidělovat i jiným uživatelům systému, kteří mu budou pomáhat spravovat systém. Tohoto může docílit v části Uživatelé této sekce. Uvidí zde přehledný seznam uživatelů, s možností jejich úpravy, pod nímž se nachází na obvyklém místě formulář s možností vložení nového uživatele systému.

Každý uživatel systému má asociováno unikátní uživatelské jméno, které je po přidání uživatele neměnné, dále pak i uživatelská práva a uživatelské heslo, které nastavuje pouze správce systému. Položka výpisu *„poslední aktivita“* ukazuje datum a čas posledního přihlášení daného uživatele.

Samotný poweruser nemůže pracovat na úkolu. Jeden zaměstnanec však může mít více účtů. Proto řešení v případě, kdyby měl jeden zaměstnanec pracovat na úkolech a taky spravovat tento systém, není problémem.

6.2.1.2 Zaměstnanec

V malých firmách často pracují zaměstnanci, jejichž počítačová gramotnost není na příliš vysoké úrovni. V praxi se proto běžně stává, že příliš složitý systém odrazuje své uživatele od jeho používání ze strachu, že se v systému nevyznají, zabloudí anebo smažou důležitá data.

Tento plánovač jde ve směru největší jednoduchosti. Proto není uživateli s právy *„zaměstnanec“* dovoleno dělat jakékoli změny v systému. Pro lidi pokročilé v znalosti práce s PC je to možná limitující faktor. Ale zato pro nově příchozí zaměstnance do firmy se tím zkrátí čas zaučení pro práci s tímto systémem na naprosté minimum.

Pro zaměstnance se stačí jen přihlásit zadáním svého uživatelského jména a hesla, a už uvidí od systému návrh práce k plnění.

Jelikož je uživatelské jméno neměnné a heslo nastavuje pouze správce systému, dává toto opatření firmě možnosti k zabezpečení přístupu zaměstnanců na tyto stránky. Např. tím, že zaměstnanci půjčí firemní přístroj, na kterém již tyto přihlašovací údaje budou uloženy a přístup z něj bude omezen pouze na tyto nebo jiné povolené stránky. Nebude pak docházet k tak častému jevu, jako jsou u zaměstnanců hesla na lístečcích po kapsách či otravná změna hesla každý měsíc. Změna hesla by mohla být prováděna centrálně. Zaměstnanec by měl jen povinnost půjčený přístroj v určitých intervalech buď vrátit do firmy, anebo pravidelně připojovat do firemní sítě, kde by tato konfigurace mohla proběhnout automaticky.

Jakmile se chce zaměstnanec přihlásit k plnění úkolu, stačí po přihlášení kliknout na tlačítko *„vzít úkol“*, čímž se mu začne odpočítávat délka provádění úkolu a od té doby bude mít možnost na svých stránkách kliknout pouze na tlačítko *„úkol splněn“*.

Při obnovení stránky dochází automaticky k obnovení rozvrhu. Toto opatření bylo zvoleno kvůli možnosti uživatele, zvyklého na volnost v rozhodování o své pracovní náplni, ovlivnit svůj rozvrh alespoň do té míry, že systému dovolí navrhnout mu jiný úkol. Mohlo by to přispět k zlepšení pracovní morálky ve firmě. V opačném případě by se dalo přistoupit k řešení, že hned prvně vygenerovaný úkol by byl zaměstnanci automaticky při přístupu na stránky přidělen, anebo by mohl být omezen počet generování rozvrhu po dokončení práce.

6.3 Plánovače práce

Evoluční algoritmy jsou náročné na výpočetní sílu serveru. O to více je nutné optimalizovat, pokud jsou počítány v prostředí skriptovacího jazyka, který se svou rychlostí rozhodně nemůže rovnat např. procedurálním jazykům jako je např. jazyk C.

Volba skriptovacího jazyku PHP je však v tomto případě dostačující. Tento plánovač je určen spíše firmám s menším počtem zaměstnanců. Kód plánovače práce je navíc vzhledem k větší optimalizaci a redukci režie nad skripty psán procedurálně, byť PHP již umožňuje objektový přístup.

Pro větší firmy by už bylo třeba algoritmus generování rozvrhu přepsat do jiného jazyka, protože vysoké počty zaměstnanců a úkolů by už vedly k řádově vyšším časovým nárokům.

PHP by tento externí program pro generování rozvrhu mohlo volat pomocí funkce *exec* jako samostatný program, anebo by mohl být připojen a volán jako dynamická knihovna přes metodu dl.

Vlastnosti genetického algoritmu plánovače práce se dají nastavovat pomocí konfiguračních globálních proměnných, které jsou uloženy v souboru *promenne.php*.

Nastavitelné jsou proměnné nutné pro obecné nastavení rozvrhu:

(9) *$rozvrh_konstanty["zacatek"]*

(10) *$rozvrh_konstanty["konec"]*

kde v pořadí první konstanta určuje začátek pracovní doby a druhá její konec. Algoritmus zajistí, aby se generovaný rozvrh vešel do limitu daného těmito proměnnými. Je proto výhodnější, aby při přidávání práce do systému byly zadávané odhady úkolů co nejpřesnější.

Dále je možno nastavit i proměnné potřebné k běžnému provozu informačního systému jako je čas k automatickému odhlášení uživatele. V případě, že by byla zvolena varianta se zapůjčením firemního přístroje zaměstnancům, mohla by být tato hodnota nastavena na relativně dlouhou dobu, aby se přístroj nemusel při každém přístupu na stránky znovu přihlašovat (pokud se uživatel explicitně neodhlásil). Tato doba se nastavuje v proměnné:

(11) *$uzprava_konstanty["delka_pripojeni"]*

Stěžejní částí tohoto konfiguračního souboru jsou konstanty používané evolučním algoritmem k tvorbě rozvrhu.

V případě, že by odhady prací byly natolik přesné, že by stačilo vygenerovat rozvrh jeden, podle něhož by se pak řídil celý zbytek dne, bylo by výhodnější nastavit tyto proměnné na vyšší hodnoty:

$evol_konstanty["jedincu_v_generaci"]
$evol_konstanty["jedincu_v_turnaji"]
$evol_konstanty["pravdepodobnost_mutace"]
$evol_konstanty["pravdepodobnost_krizeni"]
$evol_konstanty["pocet_iteraci"]

Počet jedinců v generaci označuje neměnný počet jedinců v každé iteraci výpočetní smyčky algoritmu. Čím vyšší počet jedinců, tím je větší pravděpodobnost, že algoritmus narazí na nejoptimálnější řešení. Na druhou stranu bude pak spotřebováno větší množství paměti na straně serveru.

V případě hodně vysokého počtu jedinců je nutné modifikovat soubor *php.ini* na straně serveru. V části *„Resource Limits“*, by bylo nezbytné změnit direktivu *„memory_limit“* ze standardních 128MB v závislosti na počtu jedinců.

Konfigurační proměnná *$evol_konstanty["jedincu_v_turnaji"]* určuje rychlost konvergence algoritmu. V případě vysoké hodnoty bude do turnaje vybíráno větší množství jedinců a ti s nižším fitness většinou nebudou mít možnost přistoupit k procesu křížení. Pokud bude tato proměnná nastavena na jedna, budou ke křížení připuštěni všichni jedinci aktuální generace.

Pravděpodobnosti mutaci a křížení se nevztahují na celé jedince, nýbrž na jednotlivé jejich geny. Čím nižší je tato hodnota, tím rychleji zpravidla algoritmus konverguje, i když za cenu možného horšího konečného výsledku.

Vzhledem k povaze systému a jeho zaměření na menší firmy s přibližným odhadem délky úkolů a online přístupem zaměstnanců po splnění úkolu k převzetí další práce, jsem přistoupil k řešení ukončení cyklu po určitém počtu iterací. Namísto na základě rozdílu celkového fitness ve dvou po sobě následujících generací, jehož výsledky by byly sice lepší, v tomto případě však nepotřebné.

Jakmile jsou totiž odhady prací nepřesné, je zbytečné hledat nejoptimálnější řešení z hlediska časového. Spokojíme se i s horším a rychlejším, který však také počítá s prioritami úkolů a jejich specializacemi.

Zaměstnancům i díky tomu zůstane ponechána možnost výběru práce. V rámci malé firmy budou zaměstnanci čekat na výsledek do jedné minuty (v závislosti na vytíženosti serveru), i v případě středně velkých firem je v řádu několika minut na vygenerování úkolu snesitelné.

Pokud by konstanty algoritmu byly nastaveny tak, že by opravdu docházelo k delšímu výpočtu, jsou v konfiguračním souboru na straně serveru *php.ini* v části *„Resource Limits“* nastavitelné direktivy *„max_execution_time“* a *„max_input_time“*. První jmenovaná proměnná určuje počet sekund poskytované každému skriptu na jeho dokončení. Druhá jmenovaná určuje čas, který je každému skriptu věnován na parsování dotazu. V případě přetížení serveru v důsledku dočasného náporu zaměstnanců se zájmem o zadání nové práce se dá očekávat, že by obě tyto proměnné měly být nastaveny na vyšší než standardní hodnoty.

6.3.1 Ukládání dat

Jelikož bude algoritmus plánovače pracovat s velkými objemy dat z databáze, jeho uložení musí být promyšlené. I když se bude struktura tabulek snažit procházení předcházet nutnosti procházení prvek po prvku, existují situace, kdy i sekvenční procházení pole je nevyhnutelné.

Po stažení všech informací z databáze se jednotlivé informace dle své relevance rozdělí mezi dvě tabulky. Informace, týkající se přímo úkolů či prací jsou uloženy v poli $prace_pole a informace o zaměstnancích jsou uloženy v $zamestnanci_pole.

6.3.1.1 $prace_pole

Práce pole (viz. obrázek 14) se bude nejčastěji procházet sekvenčně úkol po úkolu kvůli ruletové selekci. Proto jsou zde práce uloženy všechny v řadě po sobě. Jakmile je pseudonáhodně vybráno pořadové číslo, není nutné znovu procházet pole kvůli zjištění id práce. Ruleta navíc vybírá pouze z nedokončených a použitelných úkolů v závislosti na hodnocení, které se vypočítá z priority a počtu dnů do deadline.

Jakmile je vybráno id_práce, je druhá část pole asociativní, v níž si na jeho základě najdeme id úkolů, z nichž se skládá. V další části pole si pak podle tohoto id zjistíme podle stavu, jestli jde o práci přidělenou do virtuálního rozvrhu, který není konečný (v rámci genetického algoritmu), jestli jde o dokončený úkol, anebo jestli jde o rozpracovaný úkol. Jde-li o dokončený úkol, můžeme si také přímo zjistit id zaměstnance, komu byl přidělen.

Virtuálně přidělené úkoly jsou takové, které vidíme vygenerované v rozvrhu, ale nikdo je neplní. Může se s nimi v rámci generací jakkoli manipulovat. Aktuálně rozpracované či již dokončené práce se stávají pro algoritmus neměnné.

V poslední části tohoto pole se nachází Informace o stavu celé populace, potřebné pro správný chod algoritmu.

Pořadové číslo práce	Použitelná	
	Dokončená	
	Hodnocení	
	Id práce	
	Priorita	
	Deadline	
Id práce	Pořadí úkolu	Id úkolu
Id úkolu	Id typu	
	Specializace	
	Stav	
	Odhad	
	Id zaměstnance	
Informace	Nejkratší úkol	
	Použitelných prací	
	Použitelných úkolů	
	Dokončených prací	
	Dokončených úkolů	

Obrázek 14. Pole $prace_pole, pro ukládání informací týkajících se prací

6.3.1.2 $zamestnanec_pole

Podobně jako u $prace_pole jsou i zde informace rozděleny do několika částí. Ale jelikož se toto pole bude používat jiným způsobem, tak jak můžeme vidět na obrázku 15, jeho struktura se mírně liší. Při vybírání jednoho genu se nejdříve vylosuje práce z tabulky $prace_pole, z níž se následně vybere první použitelný úkol.

Nyní je třeba vybrat pro tento úkol zaměstnance s vhodnou specializací. Proto se v $zamestnanec_pole nejdříve vyhledává id zaměstnance podle této specializace. Pokud má vyhledávaný úkol zadaný i typ, musí se ověřit, zdali je již tento typ u nalezeného zaměstnance přítomen. Pokud ano, zjistí se z další části tabulky jeho počet provedení a průměrná délka.

Předposlední část pole obsahuje informace o celé populaci, nezbytné pro správný chod algoritmu a v té poslední jsou uloženy id zaměstnanců v řadě za sebou pro jejich rychlejší sekvenční procházení, jenž je u některých vedlejších úkonů stále nezbytné.

Id specializace	Pořadí zaměstnance	Id zaměstnance
Id zaměstnance	Jméno	
	Příjmení	
	Použitelný	
	Největší volné místo	
	Pořadí typu	Id typu
Id typu	Pořadí zaměstnance	Počet provedení
		Průměrná délka
		Id zaměstnance
Informace	Největší volné místo	
	Použitelných zaměstnanců	
Pořadové číslo Zaměstnance	Id zaměstnance	

Obrázek 15. Pole $zamestnanec_pole, pro ukládání informací týkajících se prací

6.3.2 Sestavení genetické informace

Rozvrh má přesně daný začátek (proměnná *$rozvrh_konstanty["zacatek"]*) a konec (proměnná *$rozvrh_konstanty["konec"]*) v závislosti na pracovní době zaměstnanců ve firmě. Každý zaměstnanec může mít pouze jeden rozvrh, jenž se skládá z jistého počtu úkol v závislosti na jejich délce. Algoritmus generování rozvrhu nemůže znát přesný konec úkolu, jelikož neví, kterému zaměstnanci úkol přidá, a kolik času bude potřebovat na jeho splnění. Proto systém vychází při generování z odhadů délek, určených u každého úkolu.

Při přidávání úkolu do databáze s volbou typu úkolu, systém uživateli navrhuje předpokládanou délku, jež není neměnná - konečná volba zůstává na uživateli.

Vzhledem k určení systému na generování aktuálního rozvrhu v závislosti na aktuálně dokončených úkolech a aktuální vytíženosti zaměstnanců je tato hodnota pouze orientační a je výhodnější ji nastavit na nejmenší možnou z důvodu, aby nedocházelo k zbytečnému vyřazování úkolů z rozvrhu jen kvůli tomu, že už se do něj nevejdou.

Sada rozvrhů všech zaměstnanců v pojetí evolučních algoritmů je označována jako jedinec neboli chromozóm. Jedna jeho práce je jeho gen a nemusí se vždy v jedinci vyskytovat celá. V daný den je možné dokončit jen jednu část a zbytek označit za nesplněný, aby byl buď následující den anebo někdy později dokončen.

Jako jedna generace je pak označen soubor jedinců v počtu *$evol_konstanty["jedincu_v_generaci"]*.

Před samotnou stavbou genetické informace jedince se nejdřív z databáze stáhnou veškeré nezbytné informace a uloží se do polí *$zamestnanec_pole* a *$prace_pole*. Každé z obou polí představuje jinou část databáze (struktura obou polí viz tabulka 14 a 15.)

6.3.2.1 Výběr práce pomocí rulety

Selekce z množiny volných prací probíhá ruletovou metodou. Každá práce je před náhodným výběrem ohodnocena na základě své priority a počtu dnů do deadline. Hodnocení práce probíhá podle vzorečku:

(12) *hodnocení práce = (1 / dny do deadline) * činitel*

kde činitel je konstanta, měnící se podle počtu týdnů do deadline – čím blíže je deadline, tím je vyšší. V mé implementaci algoritmu jsem zvolil dva činitele tak, aby výsledkem byla na sebe navazující posloupnost pravděpodobností úměrná počtu dnů do deadline (viz. obrázek 16).

Další zbylé týdny mají stejnou pravděpodobnost 1.

Počet dní do konce týdne	1	2	3	4	5	6	7
Ohodnocení (týden do deadline, činitel = 40)	40	20	14	10	8	7	6
Ohodnocení (dva týdny do deadline, činitel = 6)	6	3	2	2	2	1	1

Obrázek 16. Hodnocení úkolů na základě počtu dnů do deadline

6.3.2.2 Výběr zaměstnance pomocí rulety

Jakmile je práce vybrána, zjistíme si její první nedokončený úkol – jeho délku, specializaci a jestli jde o první v pořadí, nebo jestli už měl nějakého předchůdce. V případě, že předchůdce měl, zjistíme si jeho přesný konec a poznačíme si ho.

Z množiny zaměstnanců se stejnou specializací poté vybíráme opět metodou rulety člověka s dostatečně velkým místem v rozvrhu od doby dokončení předchozího úkolu.

Způsobem hodnocení zaměstnance před výběrem jsem se snažil neutralizovat nevýhodu, kterou by trpěli noví zaměstnanci ve firmě.

Pokud je hodnocen úkol bez zadaného typu, mají všichni zaměstnanci stejnou pravděpodobnost, že budou vybráni. Ale v případě, že nastoupil nový zaměstnanec, o kterém nemá systém dost informací a typ úkolu byl zadán, systém jej ohodnotí z průměrného výsledku všech ostatních zaměstnanců a tu pak ještě zprůměruje s nejlepším zaměstnancem, aby se nový zaměstnanec dostal více do čela tabulky:

(13) *Nový zaměstnanec = (celkový průměr + průměr nejlepšího) / 2*

Tím bude také více motivován k tomu, aby si své postavení udržel. Toto „zkušební období“, kdy se novým zaměstnancům nepočítá délka odpracovaného úkolu do hodnocení, je nastavitelné proměnnou *$rozvrh_konstanty["min_pocet_provedeni"]* (standardně nastavena na 10).

6.3.2.3 Sestavení prvního jedince

Už víme, jaké metody jsou použity k výběru práce a potom i zaměstnance k práci. Na konci tohoto vybíracího procesu tedy máme volného zaměstnance a úkol se stejnou specializací, jenž mu můžeme vložit do rozvrhu.

Po vložení úkolu do zaměstnancova rozvrhu a přepočítá se jeho i globální největší volné místo v rozvrhu. Následně aktualizuji počet použitelných prací (snížím o jedna) a v případě, že odhad délky práce byl roven hodnotě nejkratšího úkolu, která platí pro celou populaci, přepočítám i tuto hodnotu.

Při hledání vhodného jedince, který má i dostatečně velkou mezeru v rozvrhu na to, aby pojal řešený úkol, může nastat situace, kdy předešlý úkol končil později, než se vyskytlo dané volné místo u zaměstnance. V takovém případě se projdou i zbylí zaměstnanci s dostatečně velkým volným místem. Pokud nebude žádný z nich vyhovovat, bude práce označena za nepoužitelnou.

Sestavování prvního jedince pak probíhá do doby, než počet použitelných prací klesne na nulu (už nezbyly žádné práce na rozdávání), anebo velikost největšího místa v rozvrhu z celé populace bude menší než odhad nejmenšího úkolu (zbyly už jen úkoly, které se nikomu nevejdou do rozvrhu).

6.3.3 Vytvoření první generace jedinců

Vytvoření první generace jedinců probíhá ve funkci *vytvor_generaci*, které předchází volání funkcí *zamestnanec_stahnout* a *práce_stahnout*, které z databáze získají všechny relevantní informace a naplní jimi tabulky *$zamesntanec_pole* a *$prace_pole* způsobem, jak je popsáno výše v kapitole 7.3.1.

Poté se začne tvořit první generace jedinců kopírováním nultého jedince. V rámci tohoto se kopírují pouze nemodifikované informace, stažené už jednou z databáze. Takto předcházíme situaci, kdy by si musel každý jedinec zvlášť připojovat k databázi a zbytečně zatěžovat server, který by navíc mohl každému jedinci předat nestejné informace o dokončených úkolech.

Konečný počet jedinců v generaci, je nastavitelný proměnnou *$evol_konstanty["jedincu_v_generaci"]* v souboru *promenne.php*.

Po naplnění generace, zahájíme iterační proces mutací a křížení jedinců.

6.3.4 Hodnocení jedinců

Při každém křížení je důležité správně ohodnotit každého z jedinců. Při posuzování kvality rozvrhu jsem se rozhodl brát v úvahu hlavní čtyři faktory každého z nich. Váhu jednotlivých faktorů lze nastavit v souboru *promenne.php*:

- počet dokončených prací *($evol_konstanty["vaha_praci"]*),
- počet dokončených úkolů *($evol_konstanty["vaha_ukolu"]*),
- největší volné místo *($evol_konstanty["vaha_misto"]*),
- počet použitelných zaměstnanců *($evol_konstanty["vaha_pouzitelnych"]*)

I když by se zprvu mohlo zdát, že počet dokončených prací a úkolů by měl na hodnocení rozvrhu stačit, není tomu tak. Rozvrhy, které zahrnují větší úkoly, by byly v nevýhodě před těmi, které mají spoustu malých úkoly, i když s velkými mezerami v rozvrhu. Na druhou stranu nesmí systém upřednostňovat pouze delší úkoly před kratšími.

Tento problém je otázkou ladění při dlouhodobém používání systému a preferencích každé firmy. V případě, že by firma preferovala zhotovení celé práce před postupným dokončováním jednotlivých úkolů, může si nastavit váhu prací mnohem výše, než je váha úkolu, kterou by mohla nastavit třeba na nulu.

6.3.5 Křížení jedinců

Výběr jedinců z populace ke křížení probíhá turnajovou selekcí. Jsou tedy náhodně rozlosováni do skupin o velikost *$evol_konstanty["jedincu_v_turnaji"]* zcela bez závislosti na jejich fitness. Může díky tomu nastat situace, kdy se do jedné skupiny náhodně vyberou pouze relativně slabí jedinci. Po reprodukci se silnějším jedincem by výsledný potomek mohl představovat cestu ven z lokálního minima. Na druhou stranu se tím zpomaluje konvergence algoritmu.

Každou skupinu pak vyhrává jedinec s nejlepším fitness a následně je připojen ke stávající populaci. Jakmile budou vybráni všichni vítězové skupin, dojde ke křížení takto připojených jedinců k populaci.

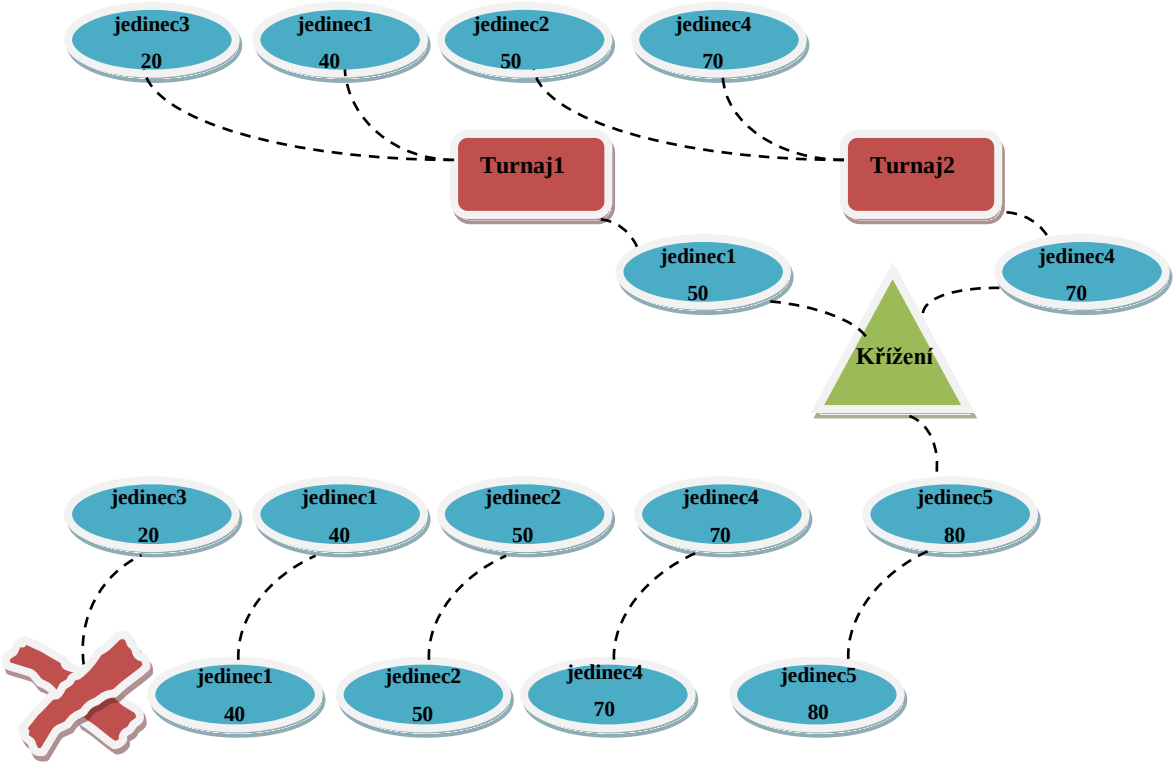
Počet jedinců ke křížení se odvíjí od počtu jedinců v turnaji a celkového počtu jedinců v generaci

(14) *Počet jedinců ke křížení = počet jedinců v generaci / počet jedinců v turnaji*

Samotný proces křížení probíhá postupným procházením jednotlivých genů v chromozómech (prací v rozvrzích obou rodičů), kdy s pravděpodobností *$evol_konstanty["pravdepodobnost_krizeni"]* dochází k mazání této genetické informace obou rodičů. Standardně se tato hodnota nastavuje na 70%, aby i v následující generaci zůstávali nejlepší jedinci z předešlé generace.

Poté geny, které byly smazány u jednoho rodiče, jsou přidávány druhému a naopak. Pokud nedojde k naplnění rozvrhu (tato situace může nastat při podobné genetické informaci obou rodičů, kdy práce, kterou by měl druhý jedinec přijmout od prvního, již obsahuje), doplní se zbylá místa použitelnými prácemi z databáze.

Rodičové v dané populaci zůstávají i po křížení. Nedochází tím pádem ke zbytečné ztrátě jedinců s potenciálně vysokým fitness při křížení s jedinci s nižším fitness.



Obrázek 16. Jedinci 1-4 jsou z první generace. Po následné turnajové selekci, křížení a následné redukci přechází jedinci 1,2,4 a 5 do další generace

6.3.6 Mutace jedinců

Ještě větší náhodnost než proces křížení do života jedinců přináší právě mutace. S nízkou pravděpodobností *$evol_konstanty["pravdepodobnost_mutace"]* (standardně nastaveno na 5%) se na konci každé generace prochází geny všech jedinců a náhodně vybraný gen se smaže. Poté se jím uvolněné místo snaží nahradit jinými, zatím nepoužitými geny z databáze.

Pravděpodobnost mutace by neměla být příliš vysoká ani nízká. Při vysoké hodnotě by docházelo k degradaci genetického materiálu jedinců a při příliš nízké hodnotě by nedocházelo ke vnášení nových genů do populace, což by znamenalo vysokou rychlost konvergence, ale také značné snížení jejich diversity.

6.3.7 Ukončení cyklu

Všeobecně se dá u evolučních algoritmů nastavit podmínka na skončení cyklu tvoření generací jako velikost změny mezi následující a předešlou generací. Výhodou oproti ukončení algoritmu po určitém počtu cyklů, který jsem použil pro tento plánovač, by byla jistota, že se algoritmus opravdu vystoupal do nějakého lokálního minima. Nevýhodou toho samého zůstává však fakt, že by zaměstnanci museli čekat na vygenerování nového rozvrhu neznámo jak dlouho.

7 Závěr

Cílem mé bakalářské práce bylo implementovat webový systém, jenž bude akceptovat několik osob a jejich pracovní zařazení a bude mít za úkol rozložit zadanou práci této skupině osob podle jejich vytíženosti.

Po nastudování evolučních algoritmů jsem se rozhodl pro tento problém použít jednu z jejich technik – genetické algoritmy.

Naimplementovaný systém je funkční a zadanou práci, která se skládá alespoň z jednoho úkolu, rozděluje mezi ostatní uživatele informačního systému.

Systém inteligentního plánování práce vedoucím ve firmách usnadňuje rozhodování o přidělování práce, usnadňuje dohled nad zaměstnanci, zlepšuje jejich vzájemnou synchronizaci a hlavně je motivuje k lepším výkonům porovnáváním s ostatními a za pomoci finančního ohodnocení.

Navíc díky pokroku ve vývoji hardware v posledních letech se online inteligentní rozdělování dá považovat za směr, který by zefektivnil chod většiny firem. Zaměstnanci by se stali mobilnějšími, samostatnějšími, rozhodovali by si o své pracovní době sami a placeni by byli opravdu jen za to, co by odpracovali. Jejich výsledky by se totiž ihned porovnávaly s ostatními spolupracovníky.

Kdyby se systém rozšířil i o zpětnou vazbu od zákazníků, měli by vedoucí i přehled nad kvalitou poskytovaných služeb svých podřízených, kteří by se ve snaze dokončit úkol co nejdříve, mohli začít chovat k zákazníkům neslušně, což by mohlo vyústit ve ztrátu klientely.

V případě, že by navíc byl tento systém vylepšen i o počítání nejoptimálnější cesty k zadaným úkolům, ušetřil by tím firmám další nemalé peníze za čas, vynaložený na plánování nejefektivnější trasy pro daný pracovní den a za neefektivní objíždění úkolů.

Zaměstnanec by měl při sobě neustále firemní bezdrátové zařízení pro přístup na internet a po každém splněném úkolu by mu systém přiděloval další práci podle toho, která by mu byla nejblíže a případně i podle její priority. Tento seznam by byl neustále aktualizován, a díky tomu by firma mohla zkrátit čekací doby zákazníků na služby. Ihned po přijmutí zakázky do systému by mohl být daný úkol poslán pracovníkovi, jenž by se vyskytoval v blízkosti zadané adresy, přímo do jeho kapesního zařízení.

Rozhodnutí systému o platovém ohodnocení jednotlivců by mělo být bráno spíše jako doporučením, které není konečné. Vedoucí jej mohou ho změnit ať již za účelem potrestání neefektivních zaměstnanců, anebo naopak za účelem zvýšit prémie efektivnímu zaměstnanci či kompenzování jím nezaviněných průtahům, jako je např. závada na firemním zařízení, dopravní kalamita při firemní cestě.

Literatura

[1] **Kvasnička V., Pospíchal J., Tiňo P., 2000:** Užitečné Evolučné Algoritmy. Tlač Vydavateľstvo STU, Bratislava: 223 s.

[2] **Schwarz J., Sekanina L., 2006**: Aplikované evoluční algoritmy. – opora k předmětu Aplikované evoluční algoritmy, FIT VUT v Brně.

[3] **W. Jason Gilmore, 2007**: Velká kniha PHP a MySQL – kompendium znalostí pro začátečníky i profesionály. Zoner Press, 864 s.

Seznam příloh

Příloha 1. CD se zdrojovými kódy informačního systému